?Simulating On-Board Compression with JPEG 2000

Janet C. Rountree[a], Brian N. Webb[a] and Michael W. Marcellin[b]
[a]Science Applications International Corporation, 101 N. Wilmot Suite 400, Tucson, AZ 85711
[b]Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721

***Abstract*- Extensive testing in a laboratory environment has demonstrated the effectiveness of the JPEG 2000 algorithm in compressing Earth Science data from a variety of orbiting instruments, including ETM+ on Landsat 7, MODIS, and Hyperion. We are now turning our attention to some of the problems to be expected when the compression is performed on board a satellite. One of these problems is the choice between compressing uncalibrated data, and performing on-board calibration. Another issue is the organization of the compressed data for downlink in such a way to minimize the distortion caused by one or more packet losses. We are also working with scientists who regularly use Earth Science data to determine the maximum compression ratio compatible with their respective applications.**

I. INTRODUCTION

JPEG 2000 is the latest International Standard for digital image compression. It provides superior image quality to the baseline JPEG standard, especially at high compression ratios, and contains many special features that facilitate its adaptation to particular types of imagery.

The compressor decides maximum image quality, up to and including lossless. Any image quality or resolution (size) can be decompressed from the resulting code stream, up to and including the maximums chosen at encode time.

For example, suppose an image is compressed losslessly. Suppose further that the resulting file is of size B_0 bytes. It is then possible to extract B_1 bytes from the file, ($B_1 < B_0$) and decompress those B_1 bytes to obtain a lossy decompressed image. This image will be identical to the image obtained if compression were performed to B_1 bytes in the first place. Similarly, it is possible to extract B_1 bytes from the file and decompress to obtain a reduced resolution image. The resulting image will be exactly the same as if the lower resolution version of the image were compressed to B_1 bytes in the first place.

In addition to the quality scalability and resolution scalability discussed above, JPEG 2000 code streams support spatial random access. There are several mechanisms to retrieve and decompress data from the code stream corresponding to selected spatial regions of an image. The different mechanisms yield different granularity of access, at varying levels of difficulty. Each region so accessed can be decoded at a variety of different resolutions and qualities.

In each case it is possible to locate, extract, and decode the bytes required for the desired image product. It is not necessary to decode the entire code stream and/or image. In many cases, the bytes extracted and decoded are identical to those that would be obtained if only the desired image products were compressed in the first place.

Many types of progressive transmission are supported by JPEG 2000. Progressive transmission is highly desirable when receiving imagery over slow communication links. As more data are received, the rendition of the displayed imagery improves in some fashion. JPEG 2000 supports progression in four dimensions: *quality*, *resolution*, *spatial location*, and *component*.

The first dimension of progressivity in JPEG 2000 is quality. As more data are received, image quality is improved.

The second dimension of progressivity in JPEG 2000 is resolution. In this type of progression, the first few bytes are used to represent a small "thumbnail" of the image. As more bytes are received, the resolution (or size) of the image increases by factors of 2 on each side. Eventually, the full size image is obtained.

The third dimension of progressivity in JPEG 2000 is spatial location. With this type of progression, imagery can be transmitted/received in approximately raster fashion, from top-to-bottom. This type of progression is particularly useful for memory-constrained decoders (such as in printers). Similarly, memory scanners (such as in space-born systems) can create spatially-progressive code streams "on-the-fly" without buffering either the image or the compressed code stream.

The fourth and final dimension of progressivity is the component. JPEG 2000 supports images with up to 16384 components. Most images with more than 4 components are from scientific instruments (e.g., Landsat). More typically, images are 1 component (grayscale), 3 components (e.g., RGB, YUV, etc.), or 4 components (CMYK). Overlay components containing text or graphics are also common. Component progression controls the order in which the data corresponding to different components is decoded. With

This work was partially supported by NASA contract NAS-00186.

progression by component, the grayscale version of an image might first be decoded, followed by color information, followed by overlaid annotations, text, etc. This type of progression, in concert with the other progression types, can be used to effect various component interleaving strategies.

II. EFFECTS OF PACKET LOSS

In this section we examine the effects of lost communication data packets on the quality of decompressed JPEG 2000 imagery. For many compression systems such loss can be catastrophic. We will demonstrate below that packet loss effects can be well contained by employing properly organized JPEG 2000 code streams. Prior to that discussion, we provide an overview of the JPEG 2000 algorithm and its built-in error resilience features.

A. The JPEG 2000 Algorithm

In JPEG 2000, an input image is first (optionally) divided into non-overlapping rectangular tiles. If the image has multiple components, an optional component transform can be applied to decorrelate these components. The samples of each component that fall into a particular tile are referred to as a tile-component. Each tile-component is then wavelet-transformed independently. After wavelet transform and quantization, each tile-component is decomposed into a set of quantization indices corresponding to certain resolution levels and associated subbands in the wavelet domain. These wavelet subbands are partitioned into several hierarchical geometric structures. The smallest structure is called a code block. Code blocks are formed by partitioning subbands at all resolution levels. Each resolution of a tile-component is partitioned into precincts. Each precinct consists of those code blocks that correspond to a particular spatial region.

Entropy coding is then performed independently on each code block using context-based, binary, arithmetic coding of bit planes. The MQ arithmetic coder is employed in this regard [1]. This coding makes three passes over each bit plane of a code block. These passes are referred to as coding passes. The compressed data from each code block can be regarded as an embedded bit stream. A code stream for the image is formed by including different numbers of coding passes from each code block. Compressed data from each precinct are arranged to form packets. A packet contains the compressed bytes from some number of coding passes from each code block in one precinct of one tile-component. One packet from each precinct of each resolution of each tile-component form a layer. Packets that belong to a particular tile are grouped together to form tile streams, and tile streams are grouped together to form a JPEG 2000 code stream.

Within a JPEG 2000 code stream, tiles, tile-parts and packets each have a header, followed by their corresponding data. A main header is inserted at the beginning of each code stream.

During the encoding procedure, a JPEG 2000 encoder typically computes a distortion-rate slope for every coding pass from every code block. This information may be used for rate allocation to control the creation of packets and layers. That being said, the standard does not specify any required rate allocation criterion or algorithm to be used. The only requirement is that a compliant decoder should be able to decode any resulting code stream. This leaves tremendous flexibility for application-specific rate allocation systems.

B. JPEG 2000 Error Resilience Features

JPEG 2000 entropy coding is based on context-based arithmetic coding. It is crucial for the arithmetic encoder and decoder to maintain synchronization in order to correctly decode a code stream. Even a single bit error in a code stream can easily destroy this synchronization, and lead to disastrous decompression effects if no special measure is taken. To combat this problem, JPEG 2000 provides some useful mechanisms for error resilience. They generally serve two purposes: (1) hierarchical data partitioning and resynchronization, (2) error detection and isolation [1].

As noted above, a JPEG 2000 code stream can be partitioned into hierarchical structures. As described below, the JPEG 2000 error resilience tools are built upon these different structures. The smallest independent coding unit is the code block. Bit errors will not propagate from one code block to another as long as code block synchronization is maintained. As mentioned above, code block data are collected into packets. A packet consists of a packet header, containing varying number of bytes from each appropriate code block. With correct packet header information, a JPEG 2000 decoder can determine the number of bytes corresponding to each code block. Thus, even though the data for a code block may be damaged (within a packet body), the decoder can reestablish synchronization with any undamaged code blocks. Similar mechanisms exist for resynchronization at the level of precincts and/or tiles.

Even when an error is confined to a single code block, it is desirable to maximize the decoded quality within that code block. To this end, JPEG 2000 provides a set of mechanisms for detecting and isolating errors to a single coding pass. All previous coding passes within the code block may then be decoded correctly.

One simple mechanism for error detection employed by JPEG 2000 is byte-stuffing. Through the use of this byte-stuffing procedure, the JPEG 2000 arithmetic coder does not produce certain values (0xFF90 through 0xFFFF) inside coding passes. Unexpected detection of one of these values in

the code stream would be noticed by a decoder to indicate that an error has occurred.

In its "default mode," a JPEG 2000 bit plane coder produces one contiguous arithmetic codeword for each code block. In this mode, a JPEG 2000 decoder may be able to detect that an error has occurred in a codeword. However, there is little hope of determining its location. In general then, the decoder has to discard the whole codeword for that code block. JPEG 2000 provides some mode variations to improve on this situation.

These modes are known as "RESTART" and "ERTERM" [1]. When the RESTART mode is used, the MQ coder is restarted at the end of each coding pass. Specifically, the MQ codeword is appropriately terminated and the MQ coder is re-initialized (excluding its probability models). Each coding pass is then represented by a separate MQ codeword segment. The length of each such segment is explicitly signaled in the relevant packet header. When the ERTERM mode is used, a predictable termination policy is employed by the encoder for each MQ codeword segment. A decoder can exploit the properties of this termination policy to detect errors which may exist in an MQ codeword segment.

When RESTART and ERTERM are used in concert, an encoder creates a separate, predictably terminated codeword segment for each coding pass. If an error occurs in the bit stream, it is very likely that a decoder will terminate in a state that is inconsistent with the predictable termination policy. Therefore, a decoder can detect the presence of errors at the end of the coding pass in which they occurred. With the length information signaled in the packet headers, the decoder can discard only the corrupt coding pass (and all subsequent coding passes from that same code block), thereby minimizing the artifacts, which result from such corruption.

C. Mitigating Packet Loss

Sophisticated techniques involving interleaving and powerful error correction codes are possible. However in this work, we take a simple approach. We assume that all error detection and or correction occurs without any specific knowledge of the JPEG 2000 code stream structure, and that data packets are either delivered correctly, or not at all. We build on our previous work in scan-based encoding to create data packets in a way that limits the damage caused by packet loss.

In previous work [2]-[4], a scan-based mode of JPEG 2000 was developed for satellite downlink compression. As mentioned previously, the basis for this mode is the careful use of precincts, and a careful implementation of the wavelet transform. As described in [1], [4] a scan-based wavelet transform can operate in an incremental fashion computing

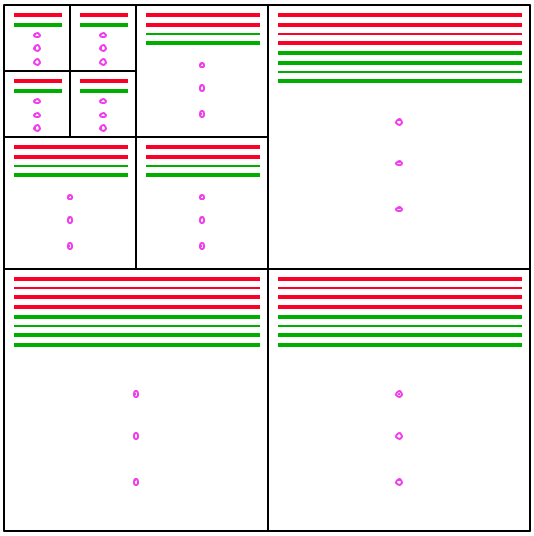

Fig. 1. Two scan elements.

the coefficients of a full-frame wavelet transform in a top-to-bottom incremental fashion without the need to buffer the entire image. The resulting coefficients can also be grouped into code blocks and precincts for compression and downlink in an incremental top-to-bottom fashion.

To this end, we created the concept of a “scan element” [4], which consists of all wavelet coefficients belonging to all precincts corresponding to a single spatial “stripe.” Fig. 1 shows the wavelet coefficients corresponding to two such scan elements, denoted by two different colors.

Fig. 2. SPOT image of Los Angeles, compressed in scan-based mode using 8 line scan elements.
Twentieth scan element is lost.

Fig. 3. SPOT image of Los Angeles, compressed in scan-based mode using 16 line scan elements. Twentieth scan element is lost.

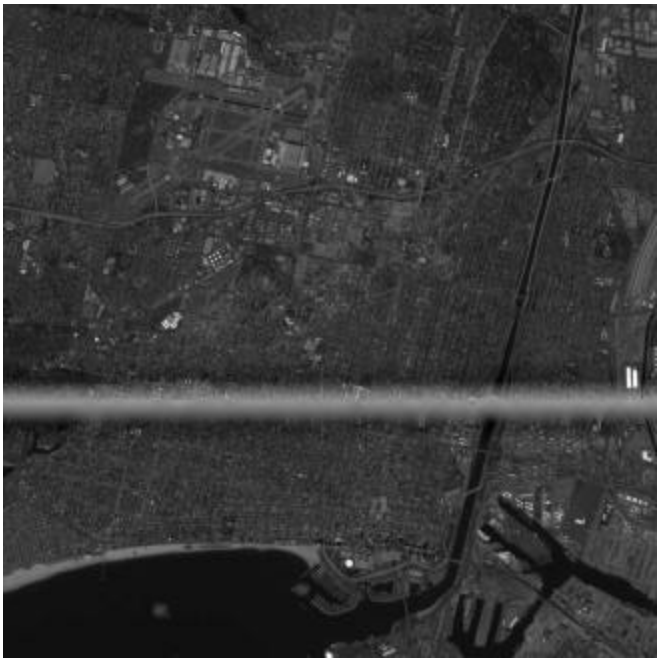

Fig. 4. SPOT image of Los Angeles, compressed in scan-based mode using 32 line scan elements. Twentieth scan element is lost.

At compression time, each scan element is initially compressed somewhat less than the target compression ratio for the image. The compressed scan element bit streams are added to a buffer at regular intervals corresponding to the frame rate. Simultaneously, data are pulled out of the buffer at some desired constant transmission rate. When the buffer is (or about to be) full, the bit streams already in the buffer along with the new bit stream to be inserted, are truncated via the embedding property to maintain constant quality across all scan elements in the buffer. This strategy relies on the highly-scalable nature of JPEG 2000.

The idea of the algorithm is that the coding passes having largest distortion-rate slopes are retained, consistent with the buffer space available. These coding passes provide the maximum possible decrease in MSE per bit spent, thus minimizing the average distortion of the frames currently in the buffer. Any coding passes that can not be accommodated in the buffer are discarded. Clearly, the discarded passes have lower slopes than those retained.

In this work, we format each scan element into a single network packet for communication purposes. We then assume that such a packet is lost and examine the effects on reconstructed image quality. Figs. 2, 3, and 4 show the effect of losing a single packet corresponding to a scan element of 8 lines, 16 lines, and 32 lines, respectively. In each case the 20^{th} scan element is lost. Thus, the damaged portion occurs further down in the image for larger scan elements.

As expected, the effects of such packet losses are well localized. The degradation is contained within a few lines above and below the lost scan element. The degradation extends above and below the scan element because of the filtering involved in performing the inverse wavelet transform during the decompression process. A theoretical analysis of this "degradation leakage" is discussed in [5], although no example imagery is provided there.

III. CALIBRATION EFFECTS

Typically, calibration is the final step in producing an image from a satellite imaging system. Calibration may be performed on the sensor platform, or resource constraints may require that this step be performed on the ground. If calibration is being performed on the ground, and if lossy compression is being used over the down-link, the compression will be performed on pre-calibrated data. This may affect compression performance and the quality of the final image data.

Fig. 5. Uncalibrated aerial1 image.

In order to quantify the effects of JPEG 2000 compression on pre- and post-calibrated data, we used the TRADES sensor simulation environment developed by Ball Aerospace. TRADES (Toolkit for Remote-sensing Analysis, Design, Evaluation and Simulation) is a software simulation of the entire remote sensing imaging system, from viewpoint simulation, illumination and atmospheric effects, imaging, detection, and calibration. TRADES supports simulation in any spectral regime from UV through passive microwave, with support for many sampling designs (whiskbroom, pushbroom, conical, step-stare scans) and many sensor types (panchromatic, filter multi-spectral, grating hyper-spectral).

We performed three calibration experiments using the TRADES environment: two simulations of a panchromatic optical sensor (aerial1 and Russia images) and one simulation of a multi-spectral sensor (similar to Landsat). For the Landsat experiment we generated high-resolution multi-spectral imagery from a 7m GSD hyper-spectral Aviris image. In each of these experiments we executed two processing runs: one in which the image was compressed after calibration, and the other in which the image was compressed prior to calibration. In each case, the final calibrated image was compared to the baseline calibrated image in which no lossy compression was performed.

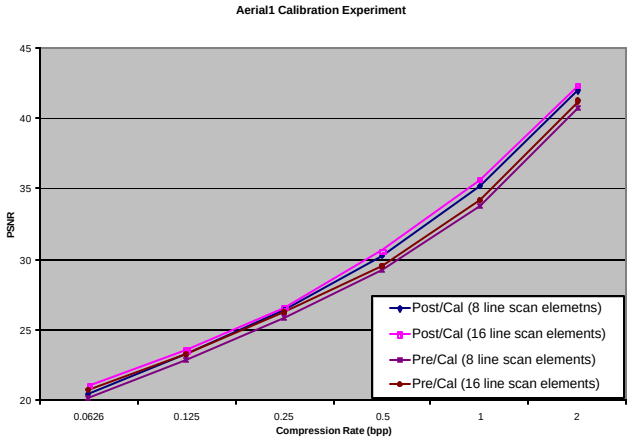

Fig. 7. Peak signal-to-noise ratio vs. bit rate for lossy compression of the aerial1 image before and after calibration.

The results were not surprising. As shown in Fig. 5, uncalibrated data tends to have horizontal discontinuities (vertical streaking) across the detector elements, and these discontinuities will tend to decrease compression performance. Lossless compression ratios decreased from 1.6/1 to 1.5/1 for the aerial1 image, from 2.0/1 to 1.7/1 for the Russia image, and from 2.0/1 to 1.4/1 for the Landsat image for calibrated and uncalibrated data, respectively.

Figs. 6, 7, and 8 show quantitative results for lossy compression performed before and after calibration. The

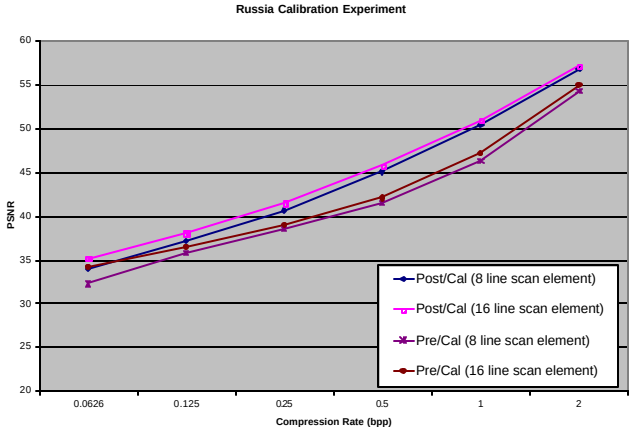

Fig. 6. Peak signal-to-noise ratio vs. bit rate for lossy compression of the Russia image before and after calibration.

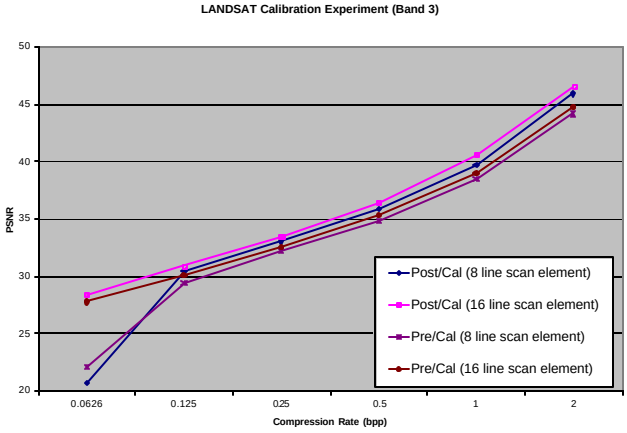

Fig. 8. Peak signal-to-noise ratio vs. bit rate for lossy compression of band 3 from the "Landsat" image before and after calibration

Fig. 9. Calibrated version of Fig. 5.

Fig. 10. Compressed/Decompressed version of Fig. 5.

results in these figures labeled “Pre/Cal” were generated by first calibrating the data then compressing/decompressing to the given bit rate. The results labeled “Post/Cal” were generated by first compressing/decompressing then calibrating.

As can be seen in these figures, lossy compression experiments showed some degradation in MSE compression performance when compressing data prior to calibration. At high bit rates (low compression ratios) the difference between Pre/Cal and Post/Cal is not readily apparent visually. However, at low rates (high compression), an interesting effect can be observed. JPEG 2000 lossy compression tends to remove some of the high-frequency components of the data. Since the calibration discontinuities are contained in the higher frequencies, the discontinuities tend to get "smoothed out" as shown in Fig. 10. Calibration then tends to add the discontinuities back into the data as shown in Fig. 11. We note here that we have chosen a particularly severe compression to ensure that the effects are readily apparent in printed form. The demonstration is clear on a CRT monitor at much less severe compression ratios.

IV. PLANNED EXPERIMENTS

Although peak signal-to-noise ratio (PSNR) is a useful indicator of the quality of a compressed/decompressed image, the real test of the success of a compression algorithm is the usefulness of the product in its intended application. In the case of panchromatic imagery, visual inspection is often enough to determine whether the desired task (e.g., feature extraction and identification) can be performed with the decompressed data. In the past, we have found that trained analysts cannot distinguish between an original image and its compressed/decompressed analogue at 2 bpp (a compression ratio of 4 to 8), and the analysts can generally perform their tasks at bit rates as low as 0.5 bpp (compression ratio of 16 to 32). Higher compression ratios are useful if rapid transmission is more important than visual quality.

In the case of multispectral (MSI) and hyperspectral (HSI) data, however, visual inspection is not sufficient to determine the utility of the compressed/decompressed version, because

Fig. 11. Calibrated version of Fig. 10.

Fig. 12 Hyperion Data Example

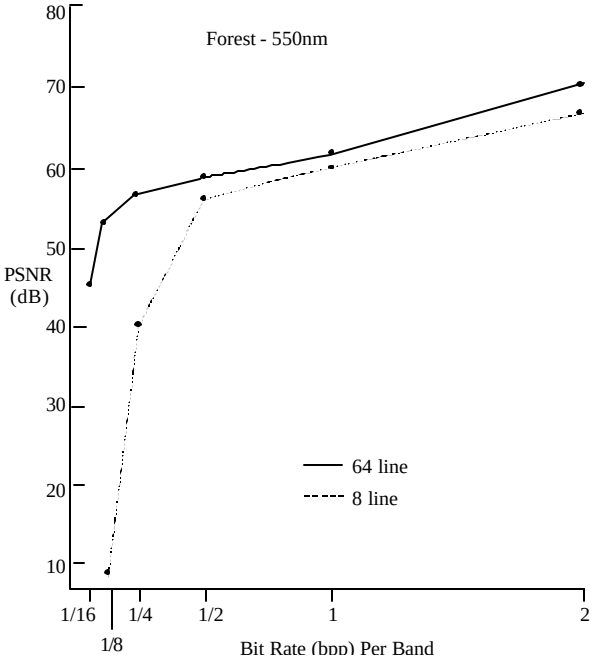

Fig. 13. Example of Hyperion Compression

the intended application usually involves mathematical computations performed by a machine, rather than visual exploitation by a human observer. The only way to test the value of a compression algorithm for this type of data is to perform the same computations on the uncompressed and on the compressed/decompressed data sets, and decide whether the difference in the results – if any – is within acceptable error limits.

With this procedure in mind, we have approached several users of hyperspectral data from the Hyperion instrument on the EO-1 satellite, to enlist their aid in evaluating the usefulness of JPEG 2000 compression for HSI. The first of these "task-oriented" experiments is being performed in collaboration with Dr. David G. Goodenough of the Canadian Forest Service, who uses AVIRIS and Hyperion data for classification and species recognition in the Greater Victoria water district. As of this writing, we have received a Hyperion data set (see Fig. 12) from Dr. Goodenough's group, and have compressed it to bit rates of 4, 2, 1, and 0.5 bpp per band (compression ratios of 4, 8, 16, and 32; note that the lossless compression ratio for this data set is 2.3). The peak signal to noise ratio for these compression experiments are shown in Fig. 13. We are awaiting the results of the scientists' evaluation.

A second experiment will involve a team of scientists working with Prof. Alfredo R. Huete of the University of Arizona's Department of Soil, Water and Environmental Science. This test, which is just getting under way, will include classification of vegetation using image statistics from Hyperion data, and seasonal profiles of vegetation indices using the lower resolution MODIS data.

In all these experiments, we must keep in mind that the test data as we receive it is not "raw" – that is, it has undergone some degree of ground-based processing, usually to Level 1. Thus, it is not completely representative of on-board compression, unless some degree of on-board pre-processing is assumed.

REFERENCES

[1] D.S. Taubman and M.W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Boston, 2002.

[2] J. C. Rountree, T. J. Flohr, M. W. Marcellin, "Report on core experiment codeff5: scan-based processing mode," ISO/IEC JTC1/SC29/WG1 N1594, March, 2000.

[3] C. Lambert-Nebout, F. Pellau, "Report on core experiment codeff5: scan-based processing mode," ISO/IEC JTC1/SC29/WG1 N1581, March, 2000.

[4] T. J. Flohr, M. W. Marcellin, J. C. Rountree, "Scan-based processing with JPEG-2000," *Proc. of SPIE, Appl. of Digital Image Proc.*, pp 347-355, July, 2000.

[5] J. Rountree, B. Webb, T. Flohr, M. Marcellin, "Optimized compression for earth science data using JPEG 2000," *Proc. Earth Science Technology Conference 2001*, NP-2001-8-338-GSFC, Greenbelt, MD, August 2001.